Boston U. Computer Science Tech. Report No. 2005-020, Aug. 2005.
To appear in *Proc. ICCV HCI Workshop*, **Oct. 2005.**

Accurate and Efficient Gesture Spotting via Pruning and Subgesture Reasoning

Jonathan Alon, Vassilis Athitsos, and Stan Sclaroff *

Computer Science Department
Boston University
Boston, MA 02215

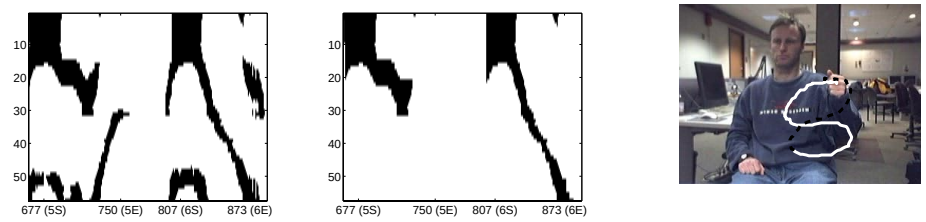
Abstract. Gesture spotting is the challenging task of locating the start and end frames of the video stream that correspond to a gesture of interest, while at the same time rejecting non-gesture motion patterns. This paper proposes a new gesture spotting and recognition algorithm that is based on the continuous dynamic programming (CDP) algorithm, and runs in real-time. To make gesture spotting efficient a pruning method is proposed that allows the system to evaluate a relatively small number of hypotheses compared to CDP. Pruning is implemented by a set of model-dependent classifiers, that are learned from training examples. To make gesture spotting more accurate a subgesture reasoning process is proposed that models the fact that some gesture models can falsely match parts of other longer gestures. In our experiments, the proposed method with pruning and subgesture modeling is an order of magnitude faster and 18% more accurate compared to the original CDP algorithm.

1 Introduction

Many vision-based gesture recognition systems assume that the input gestures are isolated or segmented, that is, the gestures start and end in some rest state. This assumption makes the recognition task easier, but at the same time it limits the naturalness of the interaction between the user and the system, and therefore negatively affects the user's experience. In more natural settings the gestures of interest are embedded in a continuous stream of motion, and their occurrence has to be detected as part of recognition. This is precisely the goal of *gesture spotting*: to locate the start point and end point of a gesture pattern, and to classify the gesture as belonging to one of predetermined gesture classes. Common applications of gesture spotting include command spotting for controlling robots [1], televisions [2], computer applications [3], and video games [4, 5].

Arguably, the most principled methods for spotting dynamic gestures are based on dynamic programming (DP) [3, 6, 7]. Finding the optimal matching between a gesture model and an input sequence using brute-force search would involve evaluating an exponential number of possible alignments. The key advantage of DP is that it can find the best alignment in polynomial time. This is

* This research was supported in part through U.S. grants ONR N00014-03-1-0108, NSF IIS-0308213 and NSF EIA-0202067.



(a) (b) (c)

Fig. 1. Pruning (a,b): example dynamic programming table for matching input stream (x axis) to a model gesture for the digit "6" (y axis). Likely observations are represented by black cells in the table (a). The cells remaining after pruning (b). In this example 87% of the cells (shown in white) were pruned. Subgesture reasoning (c): example false detection of the digit "5", which is similar to a subgesture of the digit "8".

achieved by reducing the problem of finding the best alignment to many subproblems that involve matching a part of the model to parts of the video sequence. The main novelty of our method is a pruning technique that eliminates the need to solve many of these subproblems. As a result, gesture spotting and recognition become both faster and more accurate: faster because a smaller number of hypotheses need to be evaluated; more accurate because many of the hypotheses that could have led to false matches are eliminated at an early stage. In Figure 1(b) the number of hypotheses evaluated by the proposed algorithm is proportional to the number of black pixels, and the number of hypotheses that are evaluated by a standard DP algorithm but are pruned by the proposed algorithm is proportional to the number of white pixels.

A second contribution of this paper is a novel reasoning process for deciding among multiple candidate models that match well with the current portion of the input sequence. Comparing the matching scores and using class specific thresholds, as is typically done [3, 6], is often insufficient for picking out the right model. We propose identifying, for each gesture class, the set of "subgesture" classes, i.e., the set of gesture models that are similar to subgestures of that class. While a gesture is being performed, it is natural for these subgesture classes to cause false alarms. For example, in the online digit recognition example depicted in Figure 1(c), the digit "5" may be falsely detected instead of the digit "8", because "5" is similar to a subgesture of the digit "8". The proposed subgesture reasoning can reliably recognize and avoid the bulk of those false alarms.

2 Related Work

Gesture spotting is a special case of the more general pattern spotting problem, where the goal is to find the boundaries (start points and endpoints) of patterns of interest in a long input signal. Pattern spotting has been applied to different types of input including text, speech [8], and image sequences [6].

There are two basic approaches to detection of candidate gesture boundaries: the direct approach, which precedes recognition of the gesture class, and the indirect approach, where spotting is intertwined with recognition. Methods that belong to the direct approach first compute low-level motion parameters such as velocity, acceleration, and trajectory curvature [5] or mid-level motion parameters such as human body activity [9], and then look for abrupt changes (e.g., zero-crossings) in those parameters to find candidate gesture boundaries.

In the indirect approach, the gesture boundaries are detected using the recognition scores. Most indirect methods [3, 7] are based on extensions of Dynamic Programming (DP) algorithms for isolated gestures (e.g., HMMs [10] and DTW [11]). In those methods, the gesture endpoint is detected when the recognition likelihood rises above some fixed or adaptive [3] threshold, and the gesture start point can be computed, if needed, by backtracking the optimal DP path. One such extension, continuous dynamic programming (CDP), was proposed by Oka [7]. In CDP, an input sequence is matched with a gesture model frame-by-frame. To detect a candidate gesture, the cumulative distance between them is compared to a threshold.

After a provisional set of candidates has been detected, a set of rules is applied to select the best candidate, and to identify the input subsequence with the gesture class of that candidate. Different sets of rules have been proposed in the literature: peak finding rules [6], spotting rules [12], and the user interaction model [13].

One problem that occurs in practice but is often overlooked is the false detection of gestures that are similar to parts of other longer gestures. To address this problem [3] proposed two approaches. One is limiting the response time by introducing a maximum length of the nongesture pattern that is longer than the largest gesture. Another, is taking advantage of heuristic information to catch one's completion intentions, such as moving the hand out of the camera range or freezing the hand for a while. The first approach requires a parameter setting, and the second approach limits the naturalness of the user interaction. We propose instead to explicitly model the subgesture relationship between gestures. This is a more principled way to address the problem of nested gestures, which does not require any parameter setting or heuristics.

3 Gesture Spotting

In this section we will introduce the continuous dynamic programming (CDP) algorithm for gesture spotting. We will then present our proposed pruning and subgesture reasoning methods that result in an order of magnitude speedup and 18% increase in recognition accuracy.

3.1 Continuous Dynamic Programming (CDP)

Let $M = (M_1, \ldots, M_m)$ be a model gesture, in which each M_i is a feature vector extracted from model frame i. Similarly, let $Q = (Q_1, \ldots, Q_j, \ldots)$ be a continuous

stream of feature vectors, in which each Q_j is a feature vector extracted from input frame j. We assume that a cost measure $d(i,j) \equiv d(M_i, Q_j)$ between two feature vectors M_i and Q_j is given. CDP computes the optimal path and the minimum cumulative distance $D(i,j)$ between the model subsequence $M_{1:i}$ and the input subsequence $Q_{j':j}, j' \leq j$. Several ways have been proposed in the literature to recursively define the cumulative distance. The most popular definition is:

$$D(i,j) = \min\{D(i-1,j), D(i-1,j-1), D(i,j-1)\} + d(i,j). \quad (1)$$

For the algorithm to function correctly the cumulative distance has to be initialized properly. This is achieved by introducing a dummy gesture model frame 0 that matches all input frames perfectly, that is, $D^{M^g}(0,j) = 0$ for all j. Initializing this way enables the algorithm to trigger a new warping path at every input frame.

In the online version of CDP the local distance $d(i,j)$ and the cumulative distance $D(i,j)$ need not be stored as matrices in memory. It suffices to store for each model (assuming backtracking is not required) two column vectors: the current column col_j corresponding to input frame j, and the previous column col_{j-1} corresponding to input frame $j-1$. Every vector element consists of the cumulative distance D of the corresponding cell, and possibly other useful data such as the warping path length.

3.2 CDP with Pruning (CDPP)

The CDP algorithm evaluates Eq. 1 for every possible i and j. A key observation is that for many combinations of i and j, either the feature-based distance $d(i,j)$ or the cumulative distance $D(i,j)$ can be sufficiently large to rule out all alignments going through cell (i,j). Our main contribution is that we generalize this pruning strategy by introducing a set of binary classifiers that are learned from training data offline. Those classifiers are then used to prune certain alignment hypotheses during online spotting. In our experiments, this pruning results in an order of magnitude speedup.

The proposed pruning algorithm is depicted in Algorithm 1. The input to the algorithm is input frame j, input feature vector Q_j, a set of model dependent classifiers C_i, and the previous sparse column vector. The output is the current sparse column vector.

The concept of model dependent classifiers C_i that are learned from training data offline, and are used for pruning during online spotting is novel. Different types of classifiers can be used including: subsequence classifiers, which prune based on the cumulative distance (or likelihood); transition classifiers, which prune based on the transition probability between two model frames (or states); and single observation classifiers, which prune based on the likelihood of the current observation. In our experiments we use single observation classifiers:

$$C_i(Q_j) = \begin{cases} +1 \text{ if } d(i,j) \leq \tau(i) \\ -1 \text{ if } d(i,j) > \tau(i) \end{cases}, \quad (2)$$

input : input frame j, input feature vector Q_j, classifiers C_i, and previous sparse column vector $< ind_{j-1}, list_{j-1} >$.
output: current sparse column vector $< ind_j, list_j >$.

$i = 1$;
$ptr = ind_{j-1}(0)$;
while $i \leq m$ **do**
  if $C_i(Q_j) == +1$ **then**
    nl = new element; // nl will be appended to end of $list_j$
    $nl.D = \min\{ind_j(i-1).D, ind_{j-1}(i-1).D, ind_{j-1}(i).D\} + d(i,j)$;
    $nl.i = i$;
    append($list_j$, nl);
    $ind_j = \&list_j(i)$; // & is the address-of operator, as in C
    $i = i + 1$;
  else
    //previous column empty
    if *isempty ($list_{j-1}$)* **then**
      break;
    if $ind_{j-1}(i) == NULL$ **then**
      while *ptr→next != NULL and ptr→next→i* $\leq i$ **do**
        ptr = ptr→next;
      end
      //reached the end of previous column
      if *ptr→next* $== NULL$ **then**
        break;
      i =ptr→next→i;
    else
      $i = i + 1$;
    end
  end
end

Algorithm 1: The CDPP algorithm.

where each $\tau(i)$ defines a decision stump classifier for model frame i, and is estimated as follows: the model is aligned, using DTW, with all the training examples of gestures from the same class. The distances between observation i and all the observations (in the training examples) which match observation i are saved, and the threshold $\tau(i)$ is set to the maximum distance among those distances. Setting the thresholds as specified guarantees that all positive training examples when embedded in longer test sequences will be detected by the spotting algorithm.

In order to maximize efficiency we chose a sparse vector representation that enables fast individual element access, while keeping the number of operations proportional to the sparseness of the DP table (the number of black pixels in Fig. 1(b)). The sparse vector is represented by a pair $< ind, list >$, where ind is a vector of pointers of size m (the model sequence length), and is used to reference

elements of the second variable *list*. The variable *list* is a singly linked list, where each list element is a pair that includes the cumulative distance $D(i, j)$ and the index i of the corresponding model frame. The length of *list* corresponds to the number of black pixels in the corresponding column in Fig. 1(b).

We note that in the original CDP algorithm there is no pruning, only lines 5-10 are executed inside the while loop, and i is incremented by 1. In contrast, in CDPP whenever the classifier outputs -1 and a hypothesis is pruned then i is incremented by an offset, such that the next visited cell in the current column will have at least one *active* neighbor from the previous column.

Algorithm 1 is invoked separately for every gesture model M^g. For illustration purposes we show it for a single model. After the algorithm has been invoked for the current input frame j and for all the models, the end-point detection algorithm of Sec. 3.3 is invoked.

3.3 Gesture End Point Detection and Gesture Recognition

The proposed gesture endpoint detection and gesture recognition algorithm consists of two steps: the first step updates the current list of candidate gesture models. The second step uses a set of rules to decide if a gesture was spotted, i.e., if one of the candidate models truly corresponds to a gesture performed by the user. The end point detection algorithm is invoked once for each input frame j. In order to describe the algorithm we first need the following definitions:

- **Complete path**: a legal warping path $W(M_{1:m}, Q_{j':j})$ matching an input subsequence $Q_{j':j}$ ending at frame j with the complete model $M_{1:m}$.
- **Partial path**: a legal warping path $W(M_{1:i}, Q_{j':j})$ that matches an input subsequence $Q_{j':j}$ ending at the current frame j with a model prefix $M_{1:i}$.
- **Active path**: any partial path that has not been pruned by CDPP.
- **Active model**: a model g that has a complete path ending in frame j.
- **Firing model**: an active model g with a cost below the detection acceptance threshold.
- **Subgesture relationship**: a gesture g_1 is a subgesture of gesture g_2 if it is properly contained in g_2. In this case, g_2 is a supergesture of g_1.

At the beginning of the spotting algorithm the list of candidates is empty. Then, at every input frame j, after all the CDP costs have been updated, the best firing model (if such a model exists) is considered for inclusion in the list of candidates, and existing candidates are considered for removal from the list. The best firing model will be different depending on whether or not subgesture reasoning is carried out, as described below. For every new candidate gesture we record its class, the frame at which it has been detected (or the end frame), the corresponding start frame (which can be computed by backtracking the optimal warping path), and the optimal matching cost. The algorithm for updating the list of candidates is described below. The input to this algorithm is the current list of candidates, the state of the DP tables at the current frame (the active model hypotheses and their corresponding scores), and the lists of supergestures.

The output is an updated list of candidates. Steps that involve subgesture reasoning are used in the algorithm CDPP with subgesture reasoning (CDPPS) only, and are marked appropriately.

1. Find all firing models and continue with following steps if the list of firing models is nonempty.
2. CDPPS only: conduct subgesture competitions between all pairs of firing models. If a firing model g_1 is a supergesture of another firing gesture model g_2 then remove g_2 from the list of firing models. After all pairwise competitions the list of firing models will not contain any member which is a supergesture of another member.
3. Find the best firing model, i.e., the model with the best score.
4. For all candidates g_i perform the following four tests:
   (a) CDPPS only: if the best firing model is a supergesture of any candidate g_i then mark candidate g_i for deletion.
   (b) CDPPS only: if the best firing model is a subgesture of any candidate g_i then flag the best model to not be included in the list of candidates.
   (c) If the score of the best firing model is better than the score of a candidate g_i and the start frame of the best firing model occurred after the end frame of the candidate g_i (i.e., the best firing model and candidate g_i are non-overlapping, then mark candidate g_i for deletion.
   (d) If the score of the best firing model is worse than the score of a candidate g_i and the start frame of the best firing model occurred after the end frame of the candidate g_i (i.e., the best firing model and candidate g_i are non-overlapping, then flag the best firing model to not be included in the list of candidates.
5. Remove all candidates g_i that have been marked for deletion.
6. Add the best firing model to the list of candidates if it has not been flagged to not be included in that list.

After the list of candidates has been updated then if the list of candidates is nonempty then a candidate may be "spotted", i.e., recognized as a gesture performed by the user if:

1. CDPPS only: all of its active supergesture models started after the candidate's end frame j^*. This includes the trivial case, where the candidate has an empty supergesture list, in which case it is immediately detected.
2. all current active paths started after the candidate's detected end frame j^*.
3. a specified number of frames have elapsed since the candidate was detected. This detection rule is optional and should be used when the system demands a hard real-time constraint. This rule was not used in our experiments.

Once a candidate has been detected the list of candidates is reset (emptied), and all active path hypotheses that started before the detected candidate's end frame are reset, and the entire procedure is repeated. To the best of our knowledge the idea of explicit reasoning about the subgesture relationship between gestures, as specified in steps 2, 4a, and 4b of the candidates update procedure and step 1 of the end-point detection algorithm, is novel.

Fig. 2. Palm's Graffiti digits [14].

Fig. 3. Example model digits extracted using a colored glove.

4 Experimental Evaluation

We implemented Continuous Dynamic Programming (CDP) [7] with a typical set of gesture spotting rules. In particular, we used a global acceptance threshold for detecting candidate gestures, and we used the gesture candidate overlap reasoning described in Sec. 3.3. This is the baseline algorithm, to which we compare our proposed algorithms. The proposed CDP with pruning algorithm (CDPP), is implemented as described in Sec. 3.2, with the same gesture spotting rules used in the baseline algorithm. The second proposed algorithm, CDPP with subgesture reasoning (CDPPS), includes the additional steps marked in Sec. 3.3.

We compare the baseline algorithm and the proposed algorithms in terms of efficiency and accuracy. Algorithm efficiency is measured by CPU time. Accuracy is evaluated by counting for every test sequence the number of correct detections and the number of false alarms. A correct detection corresponds to a gesture that has been detected and correctly classified. A gesture is considered to have been detected if its estimated end frame is within a specified temporal tolerance of 15 frames from the ground truth end frame. A false alarm is a gesture that either has been detected within tolerance but incorrectly classified, or its end frame is more than 15 frames away from the correct end frame of that gesture.

To evaluate our algorithm we have collected video clips of two users gesturing ten digits 0-9 in sequence. The video clips were captured with a Logitech 3000 Pro camera using an image resolution of 240×320, at a frame rate of 30 Hz. For each user we collected two types of sequences depending on what the user wore: three colored glove sequences and three long sleeves sequences; (a total of six sequences for each user). The model digit exemplars (Fig. 3) were extracted from the colored glove sequences, and were used for spotting the gestures in the long video streams. The range of the input sequence lengths is $[1149, 1699]$ frames.

The range of the digit sequence lengths is $[31, 90]$ frames. The range of the (in between digits) non-gestures sequence lengths is $[45, 83]$ frames.

For the glove sequences the hand was detected and tracked using the glove color distribution. For the other sequences the hand was detected and tracked using color and motion. A hand mask was computed using skin and non-skin color distributions [15], and was applied to an error residual image obtained by a block-based optical flow method [16]. For every frame we computed the 2D hand centroid locations and the angle between two consecutive hand locations. The feature vectors (M_i and Q_j) used to compute the local distance $d(i, j)$ are the 2D positions only. The classifier used for pruning was combination of two classifiers: one based on the 2D positions and the other based on the angle feature. Those classifiers were trained on the model digits in the offline step. To avoid overpruning we added 20 pixels to the thresholds of all position classifiers and an angle of 25 degrees to all angle classifiers.

For the end-point detection algorithm we specified the following supergesture lists that capture the subgesture relationship between digits:

Subgesture	Supergestures
"0"	{"9"}
"1"	{"4","7","9"}
"4"	{"2","5","6","8","9"}
"5"	{"8"}
"7"	{"2","3","9"}

The experimental results are summarized in Table 1. For the baseline CDP algorithm we obtained 47 correct detections and 13 false matches. For the proposed CDPP algorithm without subgesture reasoning we obtained 51 correct detections and 9 false matches, and finally for the proposed CDPP algorithm with subgesture reasoning we obtained 58 correct detections and 2 false matches. The two false matches resulted from two examples of the digit 0 that were confused as 6. Compared to CDPP without subgesture reasoning, the proposed CDPP with subgesture reasoning corrected a single instance of the digit "3" initially confused as its corresponding subdigit "7", four instances of the digit "8" initially confused as its corresponding subdigit "5", and two instances of the digit "9" initially confused as its corresponding subdigit "1".

Method	CDP	CDPP	CDPPS
Detection Rate	78.3%	85.0%	96.7%
False Matches	13	9	2

Table 1. Comparison of gesture spotting accuracy results between the baseline and the proposed gesture spotting algorithms. The accuracy results are given in terms of correct detection rates and false matches. The total number of gestures is 60.

In our experiments CDPP executed 14 times faster compared to CDP in terms of CPU time, assuming feature extraction. The overall vision-based recognition system runs comfortably in real-time.

5 Conclusion and Future Work

This paper presented a novel gesture spotting algorithm. In our experiments, this novel algorithm is an order of magnitude faster and 18% more accurate compared to continuous dynamic programming. Our current work explores other classifiers that can be used for pruning. In order to further improve our system's accuracy, we plan to incorporate a module that can make use of the DP alignment information to verify that the candidate gesture that has been detected and recognized indeed belongs to the estimated class. This is commonly known as verification in word spotting for speech [8]. Finally, rather than specifying the subgesture relationships manually we plan to learn them from training data.

References

1. Triesch, J., von der Malsburg, C.: A gesture interface for human-robot-interaction. In: Automatic Face and Gesture Recognition. (1998) 546–551
2. Freeman, W., Weissman, C.: Television control by hand gestures. Technical Report 1994-024, MERL (1994)
3. Lee, H., Kim, J.: An HMM-based threshold model approach for gesture recognition. PAMI **21** (1999) 961–973
4. Freeman, W., Roth, M.: Computer vision for computer games. In: Automatic Face and Gesture Recognition. (1996) 100–105
5. Kang, H., Lee, C., Jung, K.: Recognition-based gesture spotting in video games. Pattern Recognition Letters **25** (2004) 1701–1714
6. Morguet, P., Lang, M.: Spotting dynamic hand gestures in video image sequences using hidden Markov models. In: ICIP. (1998) 193–197
7. Oka, R.: Spotting method for classification of real world data. The Computer Journal **41** (1998) 559–565
8. Rose, R.: Word spotting from continuous speech utterances. In: Automatic Speech and Speaker Recognition - Advanced Topics. Kluwer (1996) 303–330
9. Kahol, K., Tripathi, P., Panchanathan, S.: Automated gesture segmentation from dance sequences. In: Automatic Face and Gesture Recognition. (2004) 883–888
10. Starner, T., Pentland, A.: Real-time american sign language recognition from video using hidden Markov models. In: SCV95. (1995) 265–270
11. Darrell, T., Pentland, A.: Space-time gestures. In: Proc. CVPR. (1993) 335–340
12. Yoon, H., Soh, J., Bae, Y., Yang, H.: Hand gesture recognition using combined features of location, angle and velocity. Pattern Recognition **34** (2001) 1491–1501
13. Zhu, Y., Xu, G., Kriegman, D.: A real-time approach to the spotting, representation, and recognition of hand gestures for human-computer interaction. CVIU **85** (2002) 189–208
14. Palm: Grafitti alphabet. (http://www.palmone.com/us/products/input/)
15. Jones, M., Rehg, J.: Statistical color models with application to skin detection. IJCV **46** (2002) 81–96
16. Yuan, Q., Sclaroff, S., Athistos, V.: Automatic 2D hand tracking in video sequences. In: WACV. (2005)